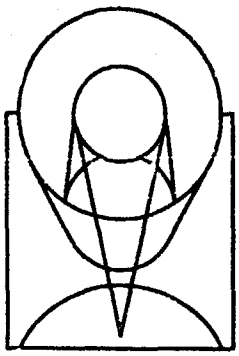

SPACE TELESCOPE SCIENCE INSTITUTE

PREPRINT SERIES
No.304

ARTIFICIAL INTELLIGENCE APPROACHES TO ASTRONOMICAL OBSERVATION SCHEDULING

Mark D. Johnston

Glenn Miller

August 1988

SPACE TELESCOPE SCIENCE INSTITUTE
3700 San Martin Drive Baltimore, MD 21218

Operated by the Association of Universities for Research in Astronomy, Inc., for the National Aeronautics and Space Administration

ARTIFICIAL INTELLIGENCE APPROACHES TO ASTRONOMICAL OBSERVATION SCHEDULING

Mark D. Johnston

Space Telescope Science Institute[1]
3700 San Martin Drive
Baltimore, MD 21218

Glenn Miller[2]

Astronomy Programs, Computer Sciences Corporation

To be published in the proceedings of the

3rd International Workshop on Data, Analysis in Astronomy

June 20 - 27, 1988

Erice, Italy (Plenum Press: NY)

[1] Space Telescope Science Institute is operated by the Association of Universities for Research in Astronomy for the National Aeronautics and Space Administration.

[2] Staff member of the Space Telescope Science Institute.

To appear in: *Proc. 3rd International Workshop on Data Analysis in Astronomy*, June 20-27 1988, Erice, Italy (Plenum Press: NY)

ARTIFICIAL INTELLIGENCE APPROACHES TO ASTRONOMICAL OBSERVATION SCHEDULING

Mark D. Johnston
Space Telescope Science Institute[1]

Glenn Miller[2]
Astronomy Programs, Computer Sciences Corporation

Abstract: Automated scheduling will play an increasing role in future ground- and space-based observatory operations. Due to the complexity of the problem, artificial intelligence technology currently offers the greatest potential for the development of scheduling tools with sufficient power and flexibility to handle realistic scheduling situations. This paper summarizes the main features of the observatory scheduling problem, how AI techniques can be applied, and recent progress on AI scheduling for Hubble Space Telescope.

1. Introduction

The purpose of automating observatory scheduling is to increase the effective utilization and, ultimately, scientific return from one or more telescopes. The development of increasingly sophisticated satellite observatories, as well as the planned high level of automation of ground-based telescopes, has led to a demand for flexible scheduling so that astronomers can optimally exploit the capabilities that these facilities have to offer.

The fundamental requirements of optimal telescope scheduling are similar in many ways to those of other scheduling problems, e.g. those encountered in commercial and industrial domains. These problems have been found to be notoriously difficult to solve in practical settings. In this paper we discuss the source of some of these difficulties and how the use of advanced software technology ("artificial intelligence", or AI) can be applied to help overcome them. We describe the progress made at Space Telescope Science Institute (STScI) in developing AI scheduling tools for Hubble Space Telescope (HST), and conclude with a discussion of the prospects for integrating automated scheduling into the overall context of observatory operations.

---

[1]Space Telescope Science Institute, 3700 San Martin Drive, Baltimore, MD 21218 USA, is operated by the Association of Universities for Research in Astronomy for the National Aeronautics and Space Administration

[2]Staff member of the Space Telescope Science Institute

2. Formulation of the Problem

The scheduling problem we are concerned with may be briefly stated as follows:

> Given a collection (pool) of *programs*, each consisting of a collection of desired observations, schedule the execution of these observations over some specified time period so that no strict constraints are violated and that preferences on when observations are scheduled, or on the conditions obtaining at these times, are satisfied to the greatest extent possible.

Hidden in this formulation is a wealth of complexity. ***Strict constraints*** refer to conditions that must not be violated under any circumstances. They determine the ***feasibility*** of a schedule. They can range from the obvious "don't observe targets too close to the sun" to more subtle statements such as "don't schedule simultaneous instrument changes on different telescopes which would overload the available operations staff." ***Preference constraints***, or simply ***preferences***, refer to conditions which are more desirable than others to hold in the final schedule. Examples of this type of constraint include scheduling observations as close to the zenith as possible, or scheduling a followup observation as soon as possible after its predecessor. Both strict and preference constraints can be expressed in terms of absolute time or relative to other observations, either past, already scheduled for the future, or part of the same scheduling pool. They can also refer to resource consumption or loading limitations.

In addition to the variety of constraints that may be relevant, other factors can complicate scheduling. Some observations may be *conditional* upon others or upon external factors: they may or may not be executed depending on whether the necessary conditions are met. Observations may be defined at differing *priority levels* which can change as the schedule evolves (e.g. it may be a high priority goal to obtain the first 80% of a statistical sample, but lower priority to obtain the last 20%.) There can also be shifts in emphasis for science reasons: targets of opportunity can disrupt the most carefully arranged schedule. Some of the most important complications are due to intrinsic uncertainty and therefore unpredictability. On the ground the weather is the most obvious such factor, but there are often many others.

3. Classical Approaches to Scheduling

Computer techniques for optimal scheduling have been investigated for many years for a number of applications (see, e.g., [1] for a comprehensive review and bibliography). Much of this classical work has focused on versions of the idealized "job-shop" scheduling problem, i.e. the problem of scheduling n tasks on m machines. This problem and related ones are NP-complete, meaning essentially that there are no efficient algorithms for finding optimal solutions (see, e.g., [2]).

The basic problem with these classical results is that they require key features of the problem to be abstracted away, so that even an "exact" solution to the abstracted problem would be of little relevance to the original "real" problem. Approximate solutions to the abstracted problem suffer from the same limitations. For example, there exist good approximate methods for finding near-optimal solutions to the well-known "travelling salesman" problem, to which the telescope scheduling problem is isomorphic if minimizing target-to-target slew time is the *sole* scheduling criterion. But this is rarely the case: other constraints enter into the problem in an essential way; these generally cannot be formulated within the framework of the abstracted problem.

It is clear that classical approaches can be useful for problems which are sufficiently simple: in practice this often means that schedule optimization is driven by a *single* overriding criterion. For the problem of scheduling complex modern observatories, however, this is not the case: more powerful techniques are required.

4. Artificial Intelligence and Scheduling

In recent years a variety of new software methodologies have been developed under the general term of "artificial intelligence" (AI). This refers to a collection of software development techniques and tools that have evolved in the course of computer science research as effective ways to represent and solve certain kinds of problems. These techniques have moved from the laboratory into widespread use in applications as their effectiveness has been demonstrated. For the purposes of automated scheduling, the most important of these are:

- a language (Lisp) that is particularly appropriate for manipulating complex data structures and symbolic data
- object oriented programming with inheritance and message passing; this facilitates the incremental development of complex problem representations
- new ways to represent and manipulate knowledge of various types, e.g. frames, associative networks, rules, demons, etc.
- methods for reasoning when facts and/or inferences are uncertain
- improved algorithms and heuristics for searching large and complex problem spaces
- integrated graphics and windows technology for facilitating user interaction

Several artificial intelligence research efforts have considered scheduling as a domain where AI techniques can be fruitfully applied. Of particular interest is the factory scheduling work of Fox, Smith, and co-workers (e.g. [3,4]) who have developed a rich constraint representation and versatile reasoning process for attacking realistic factory scheduling problems. While factory scheduling shares a number of common features with telescope scheduling (most notably a similar set of precedence and efficiency constraints), there are some important differences. Certain important factory scheduling constraints (e.g. minimizing work-in-progress or inventory) are not relevant for telescope scheduling, while the latter has a significant number of highly predictable constraints (e.g. those based on the motions of celestial objects) that can be exploited to limit the search for alternative schedules.

At Space Telescope Science Institute we initiated a project (SPIKE) in early 1987 for the purpose of developing AI scheduling tools [5,6,7] for Hubble Space Telescope (HST). HST scheduling is an extremely demanding task, requiring the scheduling of some tens of thousands of observations per year subject to a large number of proposer-specified and operational constraints [8]. Our overall approach to HST scheduling was inspired by the work of Smith and co-workers on the factory scheduling problem but has drawn on a number of other lines of research as well: as part of the SPIKE project we have developed a new framework for representing and reasoning with scheduling constraints [9] (based on discrete uncertainty reasoning for rule-based expert systems) and new techniques for searching the space of possible schedules [10] (based on recent developments in artificial neural networks).

In the following section we highlight some aspects of telescope scheduling that contribute most significantly to its complexity. This is followed by a discussion of how AI techniques and methods can be effectively applied to the problem.

5. Why is Scheduling a Hard Problem?

There are four notable features of the telescope scheduling problem that make it difficult: interacting constraints, uncertainty, optimization criteria, and search.

Interacting Constraints

As discussed above in Section 2, realistic scheduling problems will typically involve a large number of different types of constraints, both strict and preference. The first problem to overcome is that of *representation*, i.e. accurately describing the relevant constraints so that they can be interpreted by the scheduling software. A suitable representation must include not only binary (yes/no) decision criteria but must also express varying degrees of preference. The second issue is that of *trade-offs*, i.e. the knowledge of how to judge among competing or conflicting constraints. This is necessary because constraints must be considered simultaneously, not individually. The third issue can be termed *constraint reasoning*: this refers to the process of deducing, at any stage of the scheduling process, the implications of constraints and prior scheduling decisions in order to determine permitted and excluded scheduling times. This must also include possible inferences about the degree of preference of the permitted times.

It is evident that the point of constraint representation and trade-off is to capture the knowledge that human schedulers would use when faced with constructing a schedule by hand. Constraint reasoning deals with how to manipulate this knowledge dynamically as the schedule evolves in order to provide a useful view into the available scheduling choices.

Uncertainty

Because we cannot predict the future with certainty there are occasions when we cannot be sure that required preconditions for an activity will be satisfied at any given time. This unpredictability can influence scheduling in variety of ways. Unpredictable constraints can exhibit completely chaotic behavior (e.g. unexpected hardware breakdown) or can more-or-less smoothly diverge from some predicted state (e.g. a satellite position prediction). This behavior can be characterized by a *coherence time*, i.e. the timescale over which typical "significant" changes in the constraint are expected. The severity of significant changes can be qualitatively characterized by degree of *impact*, i.e. the extent to which the schedule is sensitive to changes (taking into account the trade-offs of the unpredictable constraint with any others that may be relevant). Clearly the most difficult cases to handle are short-coherence-time high-impact constraints: they may well be ignored altogether in advance scheduling and simply invoke a reaction when they occur. Constraints with coherence times that are a significant fraction of the scheduling horizon must often be incorporated explicitly.

Optimization Criteria

Since the primary purpose of automating telescope scheduling is to optimize telescope utilization, it is clearly important that a scheduling system adequately represent what is meant by "optimal". This is less straightforward than it might seem at first: scheduling goals vary depending on the circumstances, so that a schedule which is optimal in some sense can be far from

optimal in another. For example, at different times the most important optimization criterion could be some combination of overall telescope throughput, picking up a disrupted schedule, diagnosing an instrument problem, and scheduling a best match to changing environmental conditions. It is thus important that a scheduling system be flexible in terms of the high-level criteria by which schedule optimality is judged, and that *multiple* criteria be utilized depending on the circumstances.

Search

The process of scheduling can naturally be viewed as search, where at each step some decision must be made about (a) which activities to consider and (b) how to restrict their allowed scheduling times. Because of the large number of possible choices for (a) and (b) at each step, the effort required to search the space of possible schedules is typically exponential in the size of the problem. This "combinatorial explosion" is the problem most directly addressed by classical approaches to scheduling (in contrast to interacting constraints and uncertainty which are usually idealized away).

Effective search requires the early identification of both "good" decision paths as well as the early pruning of "deadend" paths, i.e. partial schedules must be judged by their *potential* for being completed beneficially as well as by their current state. This is complicated by the fact that scheduling conflicts may not be detected until many steps into the search, at which point a large amount of effort may have already been expended. It is desirable in this case to identify a minimal number of past decisions to "undo" to resolve the conflict and thus repair the partial schedule, instead of simply backing up and throwing the partial schedule away.

Another aspect of the search problem is that there may be no solution because the problem is overconstrained. Since it is generally infeasible to enumerate all possible deadends to prove that this is the case, there is a need to identify and diagnose overconstrained problems without becoming bogged down in a fruitless search.

6. AI Strategies for Optimal Scheduling

In this section we survey some of the AI techniques that can help deal with the problems described in Section 5. Many of these techniques have been implemented in the SPIKE scheduling tools; some are planned for future development.

Separate constraint reasoning from strategic search

This is a statement about the overall architecture of the scheduler. The intent is to separate those aspects of the system that reason about *constraints* from those that reason about (partial or complete) *schedules*. The reason for this separation is that these reasoning processes take place on very different levels. Constraint reasoning is low-level and determines feasible and preferred scheduling times among which choices can be made; strategic reasoning evaluates one or more schedules and actually makes the choices. There may be more than one source of strategic knowledge available to work on one scheduling problem: all would, however, make common use of the results of constraint reasoning.

Use uncertainty reasoning methods for reasoning about constraints

There has accumulated a large body of theoretical and practical results on reasoning with uncertainty in the context of discrete rule-based expert systems (e.g. Mycin [11] and Prospector

[12]). Based on this work we have developed a continuum version of uncertainty reasoning that can efficiently represent a wide variety of scheduling constraints [9]. Our framework is well-suited to the weighing of evidence for and against different scheduling hypotheses, thus providing essential inputs to trade-off decisions.

We associate with each activity (or group of activities) to be scheduled a ***suitability function***, a function of time whose value represents how desirable it is to start an activity at that time (or possibly that it is forbidden to start at that time, i.e. would violate a strict constraint). Suitability functions are derived from constraints, an arbitrary number of which may be associated with each activity depending on the type of activity and any specific factors that can affect when it is scheduled. The suitability function of an activity is the product of the suitability functions derived from its constraints. This not only mirrors an intuitive notion of how to combine different sources of evidence for and against scheduling an activity at a given time, it can also be shown to be logically required by the plausible assumptions that combination of evidence should be associative and monotonic [9].

The suitability function framework provides several important capabilities:

- a uniform way to capture human value judgements that enter into the definition of strict and preference constraints and into the trade-offs among conflicting constraints
- a straightforward mechanism for the propagation of constraints, i.e. the deduction of consequences of constraints and strategic scheduling decisions
- the explicit representation of some classes of intrinsically unpredictable constraints, in terms of maximizing the probability that desirable conditions will be met
- a mechanism to track the probable and/or certain consumption or loading of critical resources

Provide multiple control mechanisms for strategic scheduling and search

Based on the constraint-reasoning layer it is possible to implement a variety of strategic search mechanisms, any of which may be invoked depending on the nature and state of the problem. To date we have implemented three such mechanisms in the SPIKE scheduling tools:

- procedural search: this includes standard search techniques such as best-first or most-constrained-first algorithms. These tend to be computationally expensive and often encounter deadends which result in grossly sub-optimal schedules.
- rule-based heuristic search: this mechanism includes *search rules* to examine the state of a collection of partial schedules to identify the most "promising", and *commitment rules* to decide how to extend the schedule by making some scheduling decision [7]. The rules communicate with the constraint-reasoning layer through "frames" or "schema" that hold summary information about the partial schedules. This general approach is well suited to the representation of quite complex scheduling heuristics. It also has the advantage that it can be easily extended to handle new situations as they are encountered.
- neural networks: a very different approach makes use of an "artificial neural network" [13] to represent a set of discrete scheduling choices. These networks are conceptually composed of a large number of simple processing elements operating in parallel whose computational power comes from their massive interconnection. These connections can

be derived directly from the suitability functions of the activities to schedule [10]. The advantages of this approach are rapid execution, the ability to easily reschedule, and, on hardware now in the development stage, the possibility for a true parallel implementation.

Other mechanisms could also be implemented (and are currently under investigation at STScI). The most important of these are:

- plan repair mechanism: this is a facility to examine an infeasible schedule, analyze it to identify which prior decisions contributed to the conflict, and undo a sufficient number of those decisions to make the schedule feasible again. Various criteria could be employed for which types of decisions are undoable and which activities can be rescheduled. This mechanism would be useful not only for revising an infeasible schedule, but also for "reactive rescheduling" when an ongoing schedule is disrupted.

- alternative perspective focus: this mechanism would focus on certain classes of constraints (strict and preference) to help with optimizing scheduling decisions and identifying "bottlenecks" that can be determined from that class alone. For example, potential resource overloads could be detected early and then avoided by judicious scheduling of activities which use that resource. Constraints grouped into classes of this type can be regarded as corresponding to the *perspectives* of Smith et al. [4].

Formulate and attack the problem hierarchically

A common and important problem-solving strategy is to formulate and solve a simpler higher-level problem, then attack the resulting lower-level subproblems by constraining them with the higher-level solution. In the scheduling domain there are two obvious ways to accomplish this: by scheduling *groups* of related activities at once, and by limiting the *time granularity* of the schedule. For example, it is possible to cluster appropriately related activities into a single "meta-activity" which can be scheduled initially as a single entity. It is also possible to limit the initial decisions on when to schedule activities to e.g. one-week intervals out of a six-month schedule. Then, once a satisfactory allocation of activities to weeks has been determined, each week can be scheduled individually in detail.

This approach has the major advantage that some constraints which are important at the detailed scheduling level can be treated in an average or statistical sense when activities are allocated only at a sufficiently coarse-grained level. This will generally further simplify the calculation and propagation of constraints. The drawback of hierarchical scheduling is that levels may destructively interact: a deadend in a detailed schedule may require revising the higher-level schedule, which can potentially invalidate other detailed schedules. It is thus important that the higher-level problem reflect as accurately as possible the constraints that will be important in detailed scheduling.

Provide explicit user visibility and control

The approach we have taken in SPIKE is that automated scheduling is fundamentally a support tool for the people who are responsible for making scheduling decisions. In this approach one of the most important characteristics of the scheduler is how it interacts with the user. The user must have *visibility* into all aspects of the scheduling problem and the evolving schedule. The user must also have *control*, i.e. the ability to override any decisions made by the scheduler, and the ability to create and evaluate alternative schedule fragments. Because of the large volume

of information required to specify even modest-sized realistic scheduling problems, it is almost essential to utilize graphical display and interaction capabilities. This makes it necessary to implement scheduling tools of this type on single-user workstations, where high-speed graphics and dedicated processing power can both be exploited.

7. Conclusions

It is clear that software technology and approaches to scheduling have reached a sufficient level of development that automated telescope scheduling is a realistic goal. The use of artificial intelligence techniques makes it possible to develop and adapt software, such as the HST SPIKE scheduling tools, for a variety of telescope scheduling problems (see [14] for a discussion of the experimental use of SPIKE on ground-based telescope scheduling). The advantages of using these techniques are primarily a rapid software development cycle, a concise but expressive representation of scheduling data, flexibility in the definition and modification of scheduling constraints, powerful facilities for expressing search strategies, and the ability to incorporate a graphics-oriented user interface to help the user understand and modify the schedule.

Observatory scheduling is not an isolated task: for it to be ultimately successful it must be integrated into the overall operations environment. At the simplest level this integration must include the ability to inform the scheduler of what must be scheduled and what has been executed. A fuller integration should include capabilities for [15]:

- automatic access to data on environmental conditions and predictions that can be used to update scheduling constraints
- user support for proposal preparation, including observation design tools and simulators
- integrated planning capabilities, so that e.g. observations of various types are included in the scheduling pool along with appropriate calibration observations
- feedback from the schedule as executed, so that deviations and discrepancies can be recognized and diagnosed as early as possible

While this represents an ambitious program, it is not beyond the reach of current technology or its modest extrapolation.

Acknowledgements: The authors are grateful to J. Sponsler, S. Vick, K. Lindenmayer, and R. Jackson (STScI), D. Rosenthal (NASA Ames Research Center) and H.-M. Adorf (ST-ECF) for many useful discussions, and for the hospitality and support of the Space Telescope European Coordinating Facility and the European Southern Observatory (Garching) where some of this work was conducted.

References

[1] King, J.R., and Spachis, A.S. 1980: "Scheduling: Bibliography and Review," Int. Journal of Physical Distribution and Materials Management **10**, p. 105.

[2] Garey, M., and Johnson, D. 1979: *Computers and Intractability*, (W.H. Freeman & Co.: San Francisco).

[3] Fox, M, and Smith, S. 1984: "ISIS: A Knowledge-Based System for Factory Scheduling," Expert Systems **1**, p. 25.

[4] Smith, S., Fox, M., and Ow, P. 1986: "Constructing and Maintaining Detailed Construction Plans," AI Magazine, Fall 1986, p. 45

[5] Miller, G., Rosenthal, D., Cohen, W., and Johnston, M. 1987: "Expert System Tools for Hubble Space Telescope Observation Scheduling," in *Proc. 1987 Goddard Conference on Space Applications of Artificial Intelligence*; reprinted in Telematics and Infomatics **4**, p. 301 (1987).

[6] Johnston, M., 1988: "Automated Telescope Scheduling," in *Proc. Conf. on Coordination of Observational Projects*, Strasbourg, Nov. 1987, in press.

[7] Miller, G., Johnston, M., Vick, S., Sponsler, J., and Lindenmayer, K. 1988: "Knowledge Based Tools for Hubble Space Telescope Planning and Scheduling: Constraints and Strategies", in *Proc. 1988 Goddard Conference on Space Applications of Artificial Intelligence.*

[8] "HST Planning Constraints", 1987, Space Telescope Science Institute, SPIKE Report 87-1.

[9] Johnston, M. 1988: "Reasoning with Scheduling Constraints and Preferences," in preparation.

[10] Johnston, M., and Adorf, H.-M. 1988: "Scheduling with Neural Networks", in preparation.

[11] Shortliffe, E. 1987: *Computer-Based Medical Consultations: MYCIN* (American Elsevier: New York).

[12] Duda, R., Gaschnig, J., and Hart, P. 1980: "Model design in the Prospector consultant system for mineral exploration", in *Expert Systems in the Microelectronic Age*, ed. Michie, D. (Edinburgh University Press).

[13] Hopfield, J., and Tank, D. 1985: "Neural Computation of Decisions in Optimization Problems," Biological Cybernetics **52**, p. 141.

[14] Johnston, M. 1988: "Automated Observation Scheduling for the VLT", in *Proc. ESO Conference on Very Large Telescopes and their Instrumentation*, Garching, March 1988.

[15] Fosbury, R.A.E., Adorf, H.-M., and Johnston, M. 1988: "VLT Operations - the Astronomers' Environment," in *Proc. ESO Conference on Very Large Telescopes and their Instrumentation*, Garching, March 1988.